02180 Intro to AI
Exercises for week 3

Exercise 1

The following exercise is adapted from Exercise 9 in Chapter 3 of the textbook (3rd ed, Global Edition). The *missionaries and cannibals problem* is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.[1] This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

Solve the problem using the Graph-Search algorithm. You can use the pseudocode from the book or the slides. Make sure to keep track of of your *frontier* and your *explored set*. Which *search strategy* are you using (in which order do you choose frontier nodes for expansion)? Does you search strategy guarantee that you find an optimal solution (a solution with fewest possible actions)?

[1]Whenever the boat goes from one side to the other, everybody has to go out before the boat can go back again. The boat can not sail without anybody in it.

Exercise 2

Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall. When thinking about the size of the state space etc. it can be an advantage to look at concrete mazes. Consider for instance the maze and video shown here:

http://ing.dk/video/video-robotmus-overvinder-labyrint-pa-3921-sekunder-124487

a. How can the states of this problem be represented? How large is then the state space, measured as a function of the number of locations in the maze?

b. Define ACTIONS and RESULTS.

c. In navigating a maze, the only places we need to turn is at dead ends or intersections of two or more corridors (why?). Hence, from each point of the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using this observation, that is, reformulate ACTIONS and RESULTS. How large is the state space now?

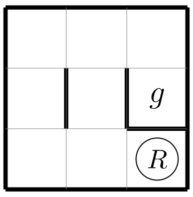
d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

e. Can you say something about the search strategy that the mouse in the video appears to be using? You are not expected to be able to give a very precise answer to this question.

f. Is finding your way through a maze like this a difficult problem in AI?

Exercise 3

In the previous exercise, we considered a robot navigating a maze. Consider the following slight variation of the problem. The robot, R, is characterised by its position only, it does not have a direction. In each state, the robot can move one cell north (N), west (W), south (S) or east (E) (unless, of course, blocked by a wall in that direction). The goal of the robot is to reach the cell marked with a g. Consider the following instance of the problem:



You should assume that the robot—and the algorithms considered below—always explore the possible moves in the following order: N, W, S, E.

a. Illustrate how BFS graph search would solve this instance of the problem. You should: 1) draw the graph generated by BFS; 2) put numbers on the nodes of the graph according to the order in which they are generated, e.g. use names $n_0, n_1, n_2, \ldots$; 3) for each iteration of the search loop, show the content of the FIFO queue used for the frontier.

b. Illustrate how DFS graph search would solve this instance of the problem. You should: 1) draw the graph generated by DFS; 2) put numbers on the nodes of the graph according to the order in which they are visited, e.g. use names $n_0, n_1, n_2, \ldots$; 3) for each iteration of the search loop, show the content of the LIFO queue (stack) used for the frontier.

c. Illustrate how DFS tree search would solve this instance of the problem. You should: 1) draw the tree generated by DFS; 2) put numbers on the nodes of the tree according to the order in which they are visited, e.g. use names $n_0, n_1, n_2, \ldots$; 3) for each iteration of the search loop, show the content of the LIFO queue (stack) used for the frontier.

d. *OPTIONAL.* Do the same as in (c) for iterative deepening DFS. This time you don't have to show each single node explored and each single configuration of the LIFO queue, but you should try to at least informally go through each step of the algorithm.

e. Could a different order of exploring the moves change the solution found by DFS? And the solution found by BFS?

f. Provide a table with an overview of the performance of the algorithms considered above in terms of: 1) number of nodes generated; 2) cost of the solution found (number of moves to get to the goal).